A Distributed Publish–Subscribe Algorithm Based on Spatial Text Information Flow

Yangtao Liu[1,*] and Xiaofeng Yu[2]

[1]*School of Computer and Software, Nanyang Institute of Technology, Nanyang 473004, China*
[2]*School of Computer and Information Technology, Nanyang Vocational College, Nanyang 474599, China*
E-mail: lyt@nyist.edu.cn; 297330152@qq.com
[*]*Corresponding Author*

Received 15 January 2023; Accepted 30 March 2023;
Publication 04 July 2023

Abstract

With the rapid development of society and the popularity of smart devices, the volume of information sent and received is increasing day by day. It has become very difficult to accurately and efficiently match a large number of events with a large number of subscriptions, and the event and subscription matching speed can no longer meet demand. To speed up the matching speed of events and subscriptions, this paper uses similarity and correlation to optimize the clustering operation and increase the data transfer and data throughput of the category fusion strategy. Firstly, the clustering operation is performed on the subscription messages, and the category to which the events belong is found according to the clustering result. Subsequently, in the category, the subscriptions matching the events are found. An on-the-fly subscription publishing algorithm is proposed to coordinate spatial information and event attribute information to handle not only the matching operation of events and subscriptions on-the-fly but also to perform subscription updates and category updates on the distributed environment on-the-fly. It can also

Journal of Web Engineering, Vol. 22_3, 385–404.
doi: 10.13052/jwe1540-9589.2231

perform clustering operations and matching operations instantly without prior knowledge. We design a distributed system for the publish–subscribe algorithm and propose a load balancing strategy for this algorithm on the distributed system. Subsequently, we experimentally validate the proposed publish–subscribe algorithm in this paper by building our own cluster and using real data.

Keywords: Publish-and-subscribe systems, distributed systems, clustering algorithms.

1 Introduction

Sending and receiving messages is a common function in many applications in real life. In the case of location services, the system pushes pertinent service information based on the user's current location; in the case of civil aviation transportation, airlines make pertinent flight plans based on air traffic control air intelligence; in wireless sensor networks [1], messages are distributed; etc. The publisher–subscribe system offers a loosely coupled message delivery and receipt architecture [2], in contrast to the traditional peer-to-peer message delivery method. In this system, the publisher posts messages to the transmission channel, subscribers submit their subscription interests, and the system automatically pushes the relevant content to the subscribers [3]. This allows for the simultaneous receipt of an event by several subscribers [4].

The major problem of the system is how to accomplish accurate and efficient matching between a high number of events and a large number of subscribers [5, 6]. The design of content-based and semantic-based publish–subscribe systems is more difficult. A bottleneck in the system will develop, which will have an impact on the real-time performance and availability of the system if the event arrival speed exceeds the event matching speed and a significant number of events cannot be transmitted in time. If a customer requests a cab using a taxi-hailing app but the order is not delivered promptly, the user experience is sure to be subpar. Many distributed systems partition data onto many servers using spatial indexes that are already in application [7].

The main contributions of this paper are as follows:

(1) This paper proposes a publish–subscribe system without a priori knowledge that can instantly process data flow and coordinate event information and spatial information.

(2) An application method that takes mixed attributes and uses the judgment parameters in the clustering algorithm to combine similarity and relevance is proposed to design and implement an instantaneous publish–subscribe algorithm on a single processor. The instantaneous clustering algorithm Rt-Cluter (RealTime-Cluter) can be used for accepted events and subscriptions on-the-fly.
(3) We design rich comparison experiments and deploy the publish–subscribe algorithm to propose a load balancing policy compatible with it, and the experiments verified the effectiveness of the publish–subscribe algorithm. The experiments verify the effectiveness of the publish–subscribe algorithm.

In Section 2, the work on the publish–subscribe system's algorithm is explained. In Section 3, the publish–subscribe system's conceptual model is formally defined. In Section 4, the real-time clustering algorithm Rt-Cluter is proposed. In Section 5, a distributed system is proposed to host the publish–subscribe algorithm. In Section 6, the results of the experiments are shown and analyzed. Section 7 concludes and looks at the work of this paper.

2 Related Work

To improve the speed of matching, researchers have proposed many new data structures to index subscriptions. A matching tree indexing structure [8] has been proposed to construct a matching tree by preprocessing the subscriptions, and when the subscriptions contain equation constraints, the matching algorithm based on the matching tree data structure is the first to realize that the matching time grows sublinearly with the number of subscriptions and the space grows linearly with the number of subscriptions. Since the same constraints in multiple subscriptions appear only once in the index structure, the matching efficiency of the matching tree is much higher than the simple method of matching one by one. The matching efficiency of a matching tree is much higher than the simple method of matching one by one. In the literature [9], the authors give precise semantics about event matching subscriptions and propose an efficient and scalable filtering algorithm based on binary decision diagrams. Each constraint in a subscription is represented as a Boolean variable, and each subscription can be represented by a binary decision graph. Based on dynamic service migration [10, 11], optimization of resource allocation [12], and time series techniques [13], when matching an event, first find all the constraints whose conditions are satisfied, and then

traverse the binary decision graph to find all matching subscriptions. There are some matching algorithms proposed based on the XML data model, for example in [14, 15].

An efficient event processing method is proposed in the literature [16]. In this method, events are represented in RDF format, and subscriptions containing matching conditions are described in SPARQL, a standard query language based on RDF format data. The Rete algorithm improves computational efficiency by preserving and sharing intermediate states and is used in the literature [17] to implement pattern matching. Based on Rete, a new matching algorithm, Treat, was designed in the literature [18] to improve the communication problems that exist when the Rete algorithm runs on parallel computers. Similar to these works, a rule engine is used to implement pattern matching in [19]. The TAMA algorithm proposed in [20] is an event-fuzzy matching and forwarding engine that reduces matching accuracy in exchange for improved matching performance. The index structure proposed in the literature [21] is dynamic and supports the addition and deletion of subscriptions. Tran and Truong argue that the cost of checking the exact coverage relationship between subscriptions is significant, so they propose reducing the cost of coverage checking by using an approximation method [22]. Approximate checking of coverage relations between subscriptions may result in omission, i.e., a new subscription is actually covered by one of the existing subscriptions but is not checked, so the new subscription continues to be forwarded. The authors argue that the omission, although leading to some redundant network transmissions, does not affect the correctness of the operation of the publish/subscribe system, and the complexity of the approximation check in time and space can be greatly reduced compared to the exact check.

3 Publish Subscription System Conceptual Model

An event or message is the term used to describe the information passed interactively between the information's producers and consumers in a publish–subscribe system [23–25], which is a middleware system that enables all participants in a distributed system to communicate asynchronously in a publish/subscribe fashion [26, 27]. In Figure 1, the conceptual model of a publish–subscribe system is depicted. Publishers (information producers) transmit events to the system, and subscribers (information consumers) submit subscription conditions to the system by specifying the events in which they are interested or by canceling their subscription if their interest

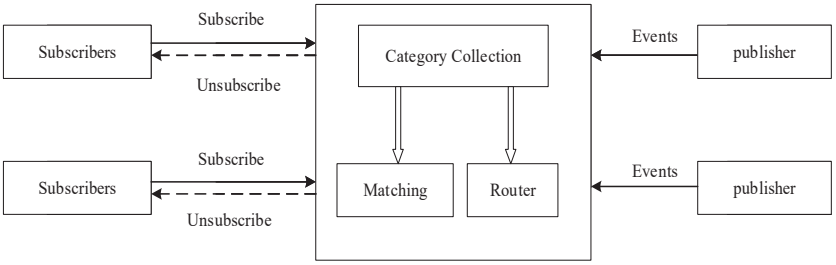

Figure 1 Schematic diagram of the basic publish-and-subscribe system.

has changed. Delivering the message published by the producer to all subscribers who have subscribed to it safely is the primary responsibility of publish–subscribe middleware.

Published events and subscriptions are two data streams that make up the publish–subscribe system. Spatial location and attribute text, which may be represented by a binary group, are the minimum number of bytes of information required for a published event to be considered in the research $(t.s, t.k)$. Where $t.s$ $t.s$ specifies the geographic location of the event, which is stated as a point in space and represented by latitude and longitude. $t.k$ stands for a collection of the event's properties. Similar to this, q designates a binary subscription message: $q.s, q.k(q.s, q.k)$, where, $q.s$ stands for the publisher's spatial range of interest for this subscription, which employs the range's smallest nearby rectangle. The information about the qualities of continuing interest to the subscriber, or $q.k$ is what is referred to as the attribute necessary for event matching.

This paper first clusters the subscriptions and creates a large number of categories $c_i(i = 1, 2, 3 \ldots)$ by clustering operations based on spatial information and event attributes, assigns them to the subscription categories for the arriving events, and then performs matching operations for the subscriptions and events in the successfully matched categories to reduce the number of unnecessary matching operations.

The event stream detects the matching category after entering the category collection, then compares each subscription in the category one at a time to identify the output that fits the subscription. If the subscription is added after it enters the category collection, the clustering method updates the category collection; if the subscription is deleted, the category to which it belongs is located and the deletion action is carried out. As subscriptions come in, the clustering algorithm gradually creates categories out of the original empty set of categories.

4 Clustering Algorithm for Publishing–Subscription Systems

The clustering method of the subscription is critical for publish–subscribe systems since it directly influences the quality of the clustering results [28], and in turn, the algorithm's overall reaction time. Researchers have suggested several anomaly detection techniques [29], including statistical test methods, grid-based methods [30], distance-based methods [31], and density-based approaches [32], to solve this issue. Due to their simplicity and effectiveness, distance-based approaches are the most popular, but they also have drawbacks such as sensitive initial center-of-mass selection, reliance on input, and poor clustering impact for non-spherical data distribution. The density-based density-based spatial clustering of applications with noise algorithm is based on the idea of grouping regions with a lot of data points into clusters [33]. Since this approach is not constrained by the shape of clusters, the accuracy of identifying noisy points is higher, and it can satisfy the demand for abnormal data detection in intelligent production lines, needed for data detection.

In this research, we present a clustering method called Rt-Cluter, which can process events and subscriptions instantly and doesn't require a training set (training is done concurrently with matching subscriptions and events). Comparative trials demonstrate this algorithm's usefulness.

Spatial information and attribute information are the two types of data that are included in subscriptions and events. In this paper, we use a hybrid approach that combines the two types of techniques. The article utilizes a grid framework to store spatial information. In this article, an inverted index structure is used for event attribute information.

Similarity and relevance serve as the evaluation factors in the Rt-Cluter algorithm. Of these, similarity mostly resolves the subscriptions clustering judgment difficulty in the clustering procedure. To allocate a new subscription q to one of the most comparable categories c, it must first be clustered based on the geographical information and attribute information of the event. The similarities between attributes and locations will be computed independently in this study and combined using a weighted total. The following is the attribute similarity KeySim formula:

$$\mathrm{KeySim} = \sum_{key \in q.k} key.pro \tag{1}$$

where key denotes the attribute and $key.pro$ denotes the proportion of attribute key in the set of attributes $c.k$ of category $c, i.e.$, the ratio of the frequency of attribute key in category c and the frequency of all attributes in category c.

Accordingly, the calculation of the spatial similarity SpatialSim requires the following equation:

$$\text{SpatialSim} = \frac{ex}{c.s} \tag{2}$$

where $c.s$ indicates the size of the area of the smallest neighboring rectangle of category c before adding subscription q, and ex indicates the size of the extension needed if category c adds subscription q, c, and s.

In conclusion, while comparing similarity, the overall similarity Sim is determined by taking both geographical and attribute similarity into account. Its mathematical formula is as follows:

$$Sim = \alpha^* KeySim/((1-\alpha) * (1 + SpatialSim)) \tag{3}$$

where α represents the weight value $(0, 1)$, which may be changed depending on the weight provided to other qualities or spaces.

The subscriptions that are awaiting assignment are put in the category with the highest similarity value. However, if the maximum similarity is still below a certain threshold, this paper creates a new category for it and assigns it to the working node where its most similar category is located, ensuring that the most similar category will be in the same working node and sparing resources in subsequent data transfer.

When the number of categories exceeds the value, the clustering algorithm additionally requires category fusion in addition to the formation of new categories. In this work, category fusion is done using correlation.

5 Distributed System Design

A single processor cannot fulfill the necessary demands as the volume of data in the network increases. In this study, we propose a distributed system to host the publish–subscribe method, and we give the algorithm a name: DRt-Cluster (Distributed-Realtime-Cluster). Figure 2 illustrates its logical structure.

The clustering node d_i and the matching node w_j are the two kinds of nodes in the system, with d_i standing for the clustering node and w_j for the matching node. The study's specifics are as follows.

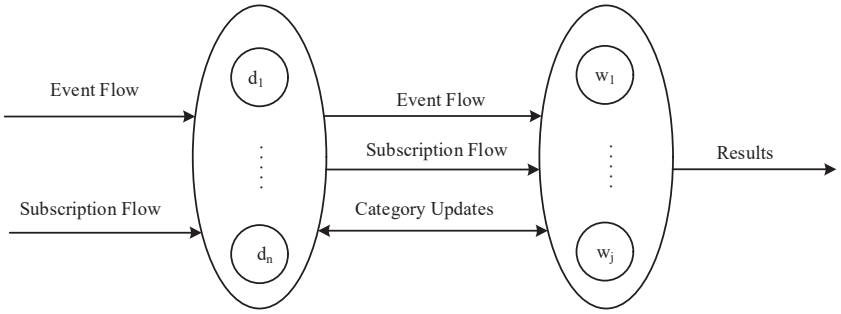

Figure 2 Logical structure of the distributed system.

Clustering node. With the following capabilities, it mostly conducts clustering operations:

The clustering node checks the properties of each category with the characteristics of event t to determine which category could have a subscription that matches. A special category is discovered for each new subscription, and the appropriate procedure is then carried out to update the category. When deleting a subscription, the category that could include the event is looked for before the deletion is carried out.

Corresponding nodes. The node keeps track of each category independently and mostly handles subscription addition and deletion as well as event matching. The following are the functions:

According to the subscription category each subscription belongs to, it locates the subscriptions for events that fit the requirements. The categories are updated following the subscription category to which they belong for subscription add/delete actions. By updating the categories in the node following the updated data of the clustering node for the category, the category update is carried out. Data transmission must be taken into consideration while implementing the method in a distributed system. To prevent overtaxing one node and creating a performance bottleneck, the load balancing issue in distributed systems should also be taken into consideration.

5.1 Category Integration

This work will prioritize the category fusion inside the same node according to the correlation, and the distributed environment needs to reduce data transmission between nodes while meeting the clustering criteria. The two variables correlate within the same worker and correlation in the different workers is recommended to be used in the procedure.

In the investigation, it was discovered that the term *c.cs* stands for the total of correlations between category c and its categories in the same matching node, and the related mathematical expression is:

$$\mathrm{c.cs} = \sum_{c \in w, c' \in w, c \neq c'} \mathrm{correlation}(c, c') \tag{4}$$

where ω stands for the node that matches the category c, which is where it is placed.

The following mathematical formula may be used to express the further finding that $c.cd$ stands for the total of the correlations between category c and the categories in each of its matching nodes:

$$\mathrm{c.cd} = \sum_{c \in w, c' \notin w} \mathrm{correlation}(c, c'). \tag{5}$$

Based on this, the judgment criterion for category c in this work is $c.MergeJudgePara$. For the fusion operation of the category with the highest relevance value inside the same matching node, the category with the biggest value is chosen. The relevant mathematical expression is expressed as follows:

$$c.MergeJudgePara = \frac{c.cs}{c.cd}. \tag{6}$$

A category fusion technique between matching nodes is suggested in this situation if a category has weak correlations with the categories in the same matching node and strong correlations with distinct matching nodes, i.e., the $MergeJudgePara$ is minimal. First, it is decided if it is appropriate to employ it just inside the node fusion by adding a new threshold merge. The category to be fused is seen to be too weakly connected with other categories inside its matching node, and only inter-node category fusion may be carried out if the maximum $MergeJudgePara$ computed is merged. This study employs inter-node category fusion, choosing the category with the smallest $MergeJudgePara$ and fusing the subscriptions within that category with subscriptions from other categories.

5.2 Load Balancing

To prevent a node from getting overloaded and acting as a performance bottleneck, load balancing's primary goal is to maintain a balance between each node's load. It may be essential to check if a node's load is balanced

after a particular number of events and subscriptions; if it is, the imbalance will be corrected as soon as it manifests.

The workload should first be measured before determining if there is a load imbalance. Following analysis, it becomes clear that category c must handle three different types of data: event t, recently added subscription q, and subscription q' that has to be removed. As a result, category c's workload is divided into the following three categories.

$$WorkLoad(c) = \int_{1}^{\text{t.num}} \text{c.size} + \sum insert(q) + \sum delete(q'). \quad (7)$$

The size of category c is indicated by the number of subscribers in the category, and $t.num$ stands for the total number of events received.

Equation (7) breaks down the workload for processing the matching operation of the event t into three parts: the first part represents the workload, where $c.size$ changes depending on whether subscriptions are added or deleted; the second part represents the resource consumption needed to add subscriptions over time; and the third part represents the resource consumption needed to delete subscriptions over time. The sum of the workloads for each category in a working node during a certain period represents the node's overall burden. The workload of matching operations is used in this study to describe the burden of categories.

This paper's load balancing method may be summed up as follows: When a node is overloaded, a portion of it is immediately transferred to another node. Assuming there are m working nodes, the total workload for all nodes in the study is set to $TotalLoad$, and if a node ω burden fulfills Equation (8),

$$WorkLoad(w) \geq \frac{2}{m} * TotalLoad. \quad (8)$$

The node's burden is thus deemed excessive and has to be balanced. The criteria used in this study to evaluate whether the transfer operation should be terminated are referenced in Equation (8). In the process of load balancing, it is required to choose a portion of the subscriptions from the node with the greatest load to transfer to other nodes.

In the study, the subset to be migrated is determined using the greedy technique, and $DiviPara$ is utilized as the selection criterion for which the operation must be carried out using the formula:

$$c.DiviPara = \frac{c.cd - c.cs}{c}.size. \quad (9)$$

Choose the category with the highest $DiviPara$ value as the subset to migrate until the load balancing requirement is satisfied. The categories inside the nodes are sorted using Equation (9), and the categories are then transferred to the nodes to be received until the load balancing need is satisfied. The node having the greatest association with the category being migrated can be chosen as the node to be received.

6 Result and Discussion

The fundamental publish–subscribe algorithm's experimental validation findings are provided in this section. The distributed environment was built on three machines. The three computers were set up using the following parameters: two PCs each with an Intel(R) Core (TM) i5-94002.8 GHz processor and 8 GB of memory, as well as one with an Intel(R) Celeron(R) CPU1007U1.5 GHz processor and 8 GB of memory. Each processor only has one core. The researcher created the distributed environment, the computer system runs Ubuntu 18.04, and the distribution system was constructed with components from ZooKeeper and Storm. Four matching nodes are involved, however, only two clustering nodes are chosen.

About 100,000 bytes of data from the website http://www.pocketgpsworld.com/ are selected to be used in the experiment. These data can only be used to create published event information, which is how this article utilizes them to create the related subscription information: first, specify the number of subscriptions, which is set to be 0.01 times the number of events in this paper; next, generate that number of subscriptions at random by selecting an event at random for each subscription, the set of subscription attributes taking a random subset of the event's attributes, and the spatial location of the subscriptions being the event. The event will be the focal point of the subscription's spatial location, which will also include a certain number of arbitrarily extended latitudes and longitude. This is done to increase the number of subscribers and events that match. The relevancy may be updated easily.

The experimental process's parameters are defined as follows in this paper: the parameter $\alpha = 0.5$ in Equation (3) balances attribute similarity and geographical similarity. The grid intervals in clustering nodes are set to 5, the grid intervals in matching nodes are set to 2, the threshold $NewTh = 0.5$ for new category generation, and the category load balancing times are set to 2, which is evenly distributed by events, i.e., if a total of 100,000 events are performed, every 50,000 are transferred to load balancing judgment.

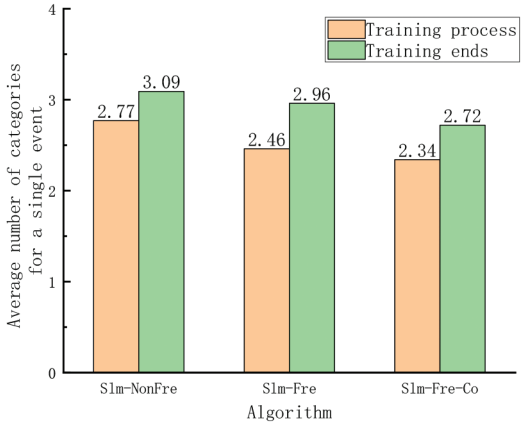

Figure 3 Average number of categories to which events are assigned.

In addition, we set the judgment threshold $merge = 1$ for category fusion within nodes and category fusion between nodes.

To increase the matching speed of events and subscriptions, first, determine whether similarity and relevance can optimize the clustering procedure. For comparison, three algorithms are offered.

(1) Perform clustering operations using $Sim - NonFre$ as a representation and solely similarity without frequency.
(2) The $Sim - Fre$ method of clustering, which uses similarity with frequency.
(3) $Sim - Fre - Co$ is used to depict the clustering procedure, which uses both similarity and correlation-containing frequency values.

The average number of categories given to the occurrences after spatial and attribute filtering is the measured data for these three methods.

(1) The process when both event and subscription streams exist, which begins at 0, or from the first event and subscription received, while the training and matching operations of the data are carried out, with the outcomes given in Figure 3.
(2) When the data training, subscription clustering, and matching processes are finished and just the event stream is present, the results are displayed in Figure 4.

Figure 3 shows the training process, where both the event stream and the subscription stream are present at the same time, and the training ends when just the event stream is being evaluated. The average number of categories for

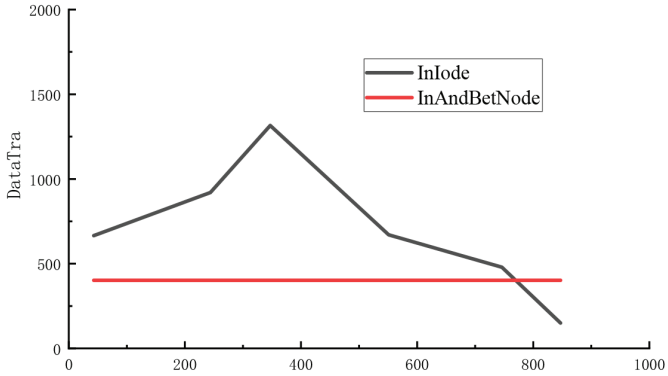

Figure 4 Category fusion data transfer volume.

which events must perform matching is highest when using $Sim - NonFre$ results, intermediate when using *Sim–Fre* results, and lowest when using $Sim - Fre - Co$ results. In other words, using similarity and correlation with frequency as opposed to simple similarity effectively reduces the number of categories for which events must perform matching operations. When simply performing matching operations after training is complete, as opposed to comparing the training process to the end of the training, the number of categories assigned to events increases because training and matching operations are carried out concurrently, and some published events that have not yet received subscription attention do so only afterward, necessitating the performance of more matching operations.

Then, the category fusion strategy's data transfer volume is confirmed. The amount of data transmission using only inside the node category fusion (using $InNode$ representation) and using both inside the node category fusion and inter-node category fusion (using $InAndBetNode$ representation), respectively, will be shown in this paper's results of $DataTra$ during the training process. In Figure 4, the experimental findings are displayed.

From the early stage of training, the data transmission utilizing both inside the node fusion and inter-node fusion $InAndBetNode$ is in a growing phase, but the categories will progressively drop after that. Because only inside the node data fusion is being used owing to the data amount in the node, the distribution of categories is not acceptable, which keeps the number of category fusion operations high.

The success of the category fusion technique will then be evaluated. The publish–subscribe method and the DRt-Cluster algorithm from this article are the two algorithms that are intended to be offered for comparison in this

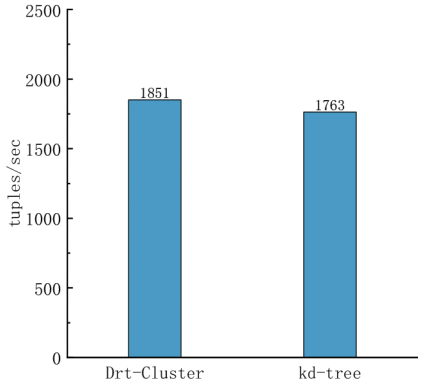

Figure 5 Throughput comparison.

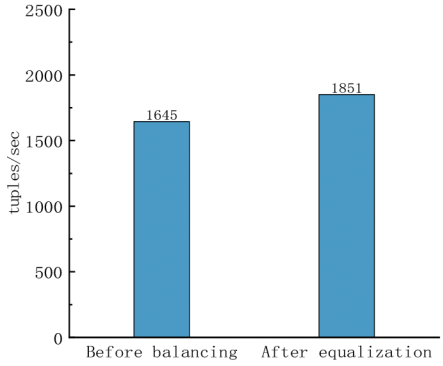

Figure 6 Throughput before and after equalization.

section. Kd-tree is used in this work to illustrate the clustering method since a study used it as the clustering algorithm. The findings of the comparison between the throughput of events in this study are displayed in Figure 5. To test the clustering results of categories and the distribution results on matching nodes using the $tran/\mathrm{sec}$ representation, the matching process of events and the addition and deletion process of subscriptions alone is carried out after the training of events and subscriptions. The throughput of the DRt-Cluster algorithm in this article is marginally higher than that of the KD-tree approach, as seen in Figure 5.

Finally, this study examines the difference in data throughput before and after the DRt-Cluster algorithm's load-balancing operation. The experimental findings are depicted in Figure 6. The distribution of categories is more

uniform and the data throughput somewhat increases after the load-balancing process.

7 Conclusion

In this paper, we first introduce the publish–subscribe system in detail, adopt the application method of mixed attributes, propose the judgment parameters in the two clustering algorithms, similarity and relevance, design and implement an instant publish-subscribe algorithm on a single processor, then design a distributed system and deploy the instant publish–subscribe algorithm DRt-Cluster on it, according to the load balancing policy that is compatible with it, and finally verify its effectiveness through experiments. Experiments demonstrate that using similarity and relevance with frequency compared to the effect of ordinary similarity effectively reduces the number of categories for which matching operations need to be performed for events, and the speed of matching events and subscriptions is improved by 15.5%.

The publish/subscribe system Is p”one ’o load balancing problems, and further research focuses on the peak hours of the research load and how the internal agent nodes can guarantee the accuracy of event matching while improving the speed of matching.

Funding Statement: This research received no external funding.

Acknowledgements: Not applicable.

Data Availability Statement: The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

Ethics Statement: Not applicable.

References

[1] T. Vaiyapuri, V. S. Parvathy, V. Manikandan, et al., “A novel hybrid optimization for cluster-based routing protocol in information-centric wireless sensor networks for IoT based mobile edge computing,” *Wireless Personal Communications*, vol. 27, no. 1, pp. 39–62, 2022.

[2] Y. Gao, F. He, S. Yu, et al., "Publish/subscribe architecture for airborne time-triggered network in avionics system," *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*, 2022, pp. 1–8.
[3] S. Kul, A. Sayar, "A survey of publish/subscribe middleware systems for microservice communication," *2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2021, pp. 781–785.
[4] A. D. Hristozov, E. T. Matson, "A methodology for estimation of software architectural complexity in publish-subscribe systems," *2022 International Conference Automatics and Informatics (ICAI)*, 2022, pp. 29–34.
[5] A. Lazidis, E. G. M. Petrakis, S. Chouliaras S, et al., "Open-source publish-subscribe systems: A comparative study. *International Conference on Advanced Information Networking and Applications*, 2022, pp. 105–115.
[6] A. Lazidis, K. Tsakos, E. G. M. Petrakis, "Publish–subscribe approaches for the IoT and the cloud: Functional and performance evaluation of open-source systems. *Internet of Things*, vol. 19, pp. 100538, 2022.
[7] I. Livaja, K. Pripužić, S. Sovilj S, et al., " A distributed geospatial publish/subscribe system on Apache Spark," *Future Generation Computer Systems*, vol. 132, pp. 282–298, 2022.
[8] A. Fertier, A. M. Barthe-Delanoë, A. Montarnal, et al., "A new emergency decision support system: the automatic interpretation and contextualisation of events to model a crisis situation in real-time," *Decision Support Systems*, vol. 133, pp. 113260, 2020.
[9] Zhang H, Zhang X, Ding K, et al., "A fuzzy matching with reasoning publish/subscribe system based on ontology," *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE). IEEE*, 2022, pp. 150–156.
[10] S. K. Pande, S. K. Panda, S. Das, "Dynamic service migration and resource management for vehicular clouds," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 1227–1247, 2021.
[11] G. Zhou, R. Zhang, S. Huang, "Generalized buffering algorithm," *IEEE Access*, vol. 9, pp. 27140–27157, 2021.
[12] J. Zhang, X. Liu, "Evaluation and optimization of QoS-aware network management framework based on process synergy and resource allocation," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–9, 2018.

[13] H. Abbasimehr, M. Shabani, "A new framework for predicting customer behavior in terms of RFM by considering the temporal aspect based on time series techniques," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 1, pp. 515–531, 2021.

[14] R. Chen, Z. Wang, Y. Hong, "Pipelined XPath query based on cost optimization," *Scientific Programming*, vol. 2021, pp. 1–16, 2021.

[15] B. Cao, Y. Gu, Z. Lv, et al., "RFID reader anticollision based on distributed parallel particle swarm optimization," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3099–3107, 2021.

[16] M. Rinne, E. Nuutila, "User-configurable semantic data stream reasoning using SPARQL update," *Journal on Data Semantics*, vol. 6, no. 3, pp. 125–138, 2017.

[17] C. Huang, C. Zhang, J. Zhao, et al., "Mist: A multiview and multimodal spatial-temporal learning framework for citywide abnormal event forecasting," *The World Wide Web Conference*, 2019, pp. 717–728.

[18] F. Mo, H. U. Rehman, F. M. Monetti, et al., "A framework for manufacturing system reconfiguration and optimisation utilising digital twins and modular artificial intelligence," *Robotics and Computer-Integrated Manufacturing*, vol. 82, pp. 102524, 2023.

[19] Durkovic S, Cica Z. Multicast Load-Balanced Birkhoff-Von Neumann Switch With Greedy Scheduling. IEEE Access, 2020, 8, pp. 120654–120667.

[20] M. Florea, C. Potlog, P. Pollner, et al., "Complex project to develop real tools for identifying and countering terrorism: real-time early detection and alert system for online terrorist content based on natural language processing, social network analysis, artificial intelligence and complex event processing," *Challenges in Cybersecurity and Privacy-the European Research Landscape*, River Publishers, 2022, pp. 181–206.

[21] I. Livaja, K. Pripužic, S. Sovilj, et al., "A distributed geospatial publish/subscribe system on Apache Spark," *Future Generation Computer Systems*, vol. 132, pp. 282–298, 2022.

[22] D. A Tran, L. H. Truong, "Enabling publish/subscribe services in sensor networks," *Future Internet Services and Service Architectures*. River Publishers, 2022, pp. 339–363.

[23] V. Rampérez, J. Soriano, D. Lizcano, et al., "Automatic evaluation and comparison of pub/sub systems performance improvements," *Journal of Web Engineering*, pp. 1055–1080, 2022.

[24] M. Nast, H. Raddatz, B. Rother, et al., "A survey and comparison of publish/subscribe protocols for the Industrial Internet of Things (IIoT),"

Proceedings of the 12th International Conference on the Internet of Things, 2022, pp. 193–200.

[25] R. Van Glabbeek, D. Deac, T. Perale, et al., "Flexible and efficient security framework for many-to-many communication in a publish/subscribe architecture," *Sensors*, vol. 22, no. 19, p. 7391, 2022.

[26] C. Miguel, V. Rampérez, J. Soriano, et al., "Towards SLA-driven autoscaling of cloud distributed services for mobile communications," *Mobile Information Systems*, vol. 2022, 2022.

[27] C. Prajisha, A. R. Vasudevan, "An efficient intrusion detection system for MQTT-IoT using enhanced chaotic salp swarm algorithm and LightGBM," *International Journal of Information Security*, vol. 21, no. 6, pp. 1263–1282, 2022.

[28] M. Cheng, T. Ma, L. Ma, et al., "Adaptive grid-based forest-like clustering algorithm," *Neurocomputing*, vol. 481, pp. 168–181, 2022.

[29] W. Zhu, Y. Deng, S. Qian, et al., "PEM: A parallel ensemble matching framework for content-based publish/subscribe systems," *The 34th International Conference on Software Engineering and Knowledge Engineering*, 2022.

[30] M. Shahbazi, M. Simsek, B. Kantarci, "Density-based clustering and performance enhancement of aeronautical ad hoc networks," *2022 International Balkan Conference on Communications and Networking (BalkanCom)*, 2022, pp. 51–56.

[31] G. Wu, L. Cao, H. Tian, et al., "HY-DBSCAN: A hybrid parallel DBSCAN clustering algorithm scalable on distributed-memory computers," *Journal of Parallel and Distributed Computing*, vol. 168, pp. 57–69, 2022.

[32] V. Rampérez, J. Soriano, D. Lizcano, et al., "FLAS: A combination of proactive and reactive auto-scaling architecture for distributed services," *Future Generation Computer Systems*, vol. 118, pp. 56–72, 2021.

[33] T. Ouyang, X. Shen, "Online structural clustering based on DBSCAN extension with granular descriptors," *Information Sciences*, vol. 607, pp. 688–704, 2022.

Biographies

Yangtao Liu received his master's degree in Computer Technology from Huazhong University of Science and Technology, China, in 2010. He joined Nanyang Institute of Technology, China, in 2004, where he is currently a lecturer. His research interests include big data technology and software engineering.

Xiaofeng Yu received his bachelor's degree in computer science and technology from Nanyang Institute of Technology, China, in 2008 and his master's degree in education administration from Krirk University, Thailand, in 2021. He joined Nanyang Vocational College in 2011. His research interest is software engineering.